FORMULARIO DE PRESENTACIÓN DE CURSOS DE POSGRADO

1.1. Indique la denominación del curso propuesto:

Programación en Python.

1.2. Inserto en un carrera de posgrado

Sí ✘ No

1.3. En caso de que el curso ya sea dictado en otra carrera indique la siguiente información:

Carrera	Tipo de dictado	Modalidad	Carácter
Lic. en Ciencias Básicas de la FCEN	Semanal	virtual debido a la situación sanitaria	electivo

2. Equipo docente.

2.1. Responsable a cargo.

Apellido: Millán
Nombre: Emmanuel Nicolas
Documento: DNI 29974940
Correo electrónico: emmanuel.millan@itu.uncu.edu.ar
CUIT/CUIL: 20-29974940-2

2.2. Integrantes del equipo docente (repetir cuantas veces sea necesario)

Apellido…….
Nombre:……
Documento:……..
Correo electrónico:……
CUIT/CUIL:……

3. Fecha probable de dictado

Semestre 1er ✘ 2do mes: Abril/Junio

4. Número máximo y mínimo de alumnos

Máximo 10

5. Carga horaria propuesta:

90 horas (60 horas de cursado teórico-práctico y 30 horas de elaboración de un proyecto integrador)

5.1. Exprese la carga horaria relacionada al dictado de la actividad en horas reloj.

Modalidad	Carga teórica	Carga práctica	Total	Porcentaje
Presencial	20	40	60	66
No presencial	0	30	30	33
Total	20	70	90	

Debido a la situación sanitaria la totalidad de la carga horaria se realizará a distancia.

6. Objetivos

1. Dominar los conceptos básicos del lenguaje Python.
2. Identificar y organizar los datos pertinentes al problema, evaluando el contexto particular.
3. Comprender las ventajas y desventajas del lenguaje de Python.
4. Utilizar conceptos básicos de complejidad computacional en Python.
5. Conocer los diversos paquetes científicos disponibles para Python para funcionalidad avanzada.
6. Conocer las fuentes de información disponibles para el usuario para seguir mejorando sus habilidades.
7. Resolver ejercicios prácticos sencillos y específicos utilizando códigos propios en lenguaje Python

7. Contenidos. (2000 caracteres)

Unidad 1: Introducción general
Introducción general a Python. Sintaxis del lenguaje. Lógica binaria. Lenguajes de programación. Introducción a los cuadernos Jupyter y la programación interactiva. Peculiaridades de Python. Control de paquetes.

Unidad 2: Resolución de problemas
Problemas y algoritmos. Conceptos, características y clasificación de algoritmos. Partes y representación. Clasificación de problemas y etapas para su resolución. Estructuración de datos. Constantes y variables. Entrada, proceso y salida. Tipos de datos: numéricos, lógicos y de caracteres.

Unidad 3: Diagramación
Pseudocódigo. Características de un lenguaje estructurado. Diagramas de Flujo. Elementos, reglas y usos. Sentencias secuenciales, condicionales y repetitivas. Modularización de algoritmos. Concepto de funciones y procedimientos.

Unidad 4: Estructuras de datos
Arreglos, diccionarios, listas, y tuplas. Carga, búsqueda, y muestra de datos. Operaciones con estructuras de datos. Métodos de ordenamiento, búsqueda e intercalación.

Unidad 5: Entorno de desarrollo
Estructura de un programa. Bibliotecas de funciones. Tipos de datos. Operadores. Funciones de entrada y salida de datos. Asignación. Estructuras de control de flujo del programa. Entrada y salida de archivos de texto, o archivos de datos.

Unidad 6: Funciones en Python
Modularización de programas. Funciones: declaración, llamado y definición de funciones en Python. Paso de argumentos a una función. Variables. Alcance de las variables: locales y globales.

Unidad 7: Análisis de datos científicos - Aplicaciones
Introducción y aplicación del ecosistema SciPy y sus varios paquetes para trabajo con datos científicos. Aplicación del ecosistema a ejemplos prácticos de conjuntos de datos empíricos en distintas áreas de ciencia. Revisión de bibliotecas como Numpy, Pandas, scikit-image y scikit-learn.

Unidad 8: Visualización de datos
Introducción al paquete Matplotlib. Trazar gráficos unidimensionales, y multidimensionales. Producción de gráficos de calidad profesional. Aplicación del paquete Matplotlib a ejemplos en diferentes áreas de la ciencia.

8. Describa las actividades prácticas desarrolladas, indicando lugar donde se desarrollan y modalidad de supervisión. (Si corresponde). (2000 caracteres)

Las actividades prácticas se llevarán a cabo durante el horario de clase programado, en conjunto con las secciones teóricas. Actualmente se prevé que el trabajo práctico representará aproximadamente el 60% de las clases programadas, aunque este número puede variar de una clase a otra.

Las actividades prácticas tomarán la forma de ejemplos guiados utilizando los cuadernos interactivos Jupyter. De este modo, permite a los asistentes probar sus habilidades de programación a través de una interfaz más fácil de usar. Las evaluaciones de los asistentes también tomarán la forma de cuadernos interactivos y se discuten con mayor detalle en la sección diez de este documento.

Como las actividades prácticas se llevarán a cabo durante la clase y utilizando los mismos materiales (es decir, computadoras, lápiz y papel), se llevarán a cabo en el mismo espacio que la clase teórica. Por lo tanto, no se requerirá supervisión adicional de los asistentes. En caso de ser necesario por la situación sanitaria, el espacio curricular está preparado para ser dictado 100% virtual

9. Bibliografía propuesta

1. C. Horstmann and R. Necaise, Python for everyone, Wiley, Hoboken, New Jersey, 1st edn., 2014.
2. Scipy-cookbook, https://scipy-cookbook.readthedocs.io. (accedido Noviembre 2020)
3. Matplotlib, https://matplotlib.org/index.html. (accedido Febrero 2020)
4. NumPy v1.19 Manual, https://docs.scipy.org/doc/numpy/. (accedido Noviembre 2020)
5. D. Beazley and B. Jones, Python Cookbook, O'Reilly, Sebastopol California, 3rd edn., 2013.
6. J. VanderPlas, Python Data Science Handbook, O'Reilly Media, Sebastopol California, 1st edn., 2016.
7. W. McKinney, Python for data analysis, O'Reilly, Sebastopol California, 1st edn., 2012.
8. SymPy v1.6.2, https://docs.sympy.org/latest/index.html. (accedido Noviembre 2020)
9. Pandas v1.1.4, http://pandas.pydata.org/pandas-docs/stable/. (accedido Noviembre 2020)
10. S. Tosi, Matplotlib for Python developers, Packt Pub., Birmingham, U.K., 1st edn., 2009.

10. Modalidad de evaluación y requisitos de aprobación y promoción.

Los asistentes tendrán una evaluación integradora al final del cursado y deberán entregar un trabajo final integrador. La evaluación integradora tomará la forma de ejercicios computacionales, en los cuales los asistentes deberán crear o completar

programas informáticos en Python para resolver problemas específicos. El acceso al material del curso e internet no se restringirá durante las evaluaciones, ya que refleja un entorno de trabajo realista. Los ejercicios enviados se evaluarán en función de su capacidad para resolver el problema, la eficiencia del código, la legibilidad del código y la calidad de salida (cuando corresponda). El trabajo final de investigación se enfocará en la utilización de técnicas avanzadas de programación en Python, por lo que se requerirá un desarrollo mayor y el uso de técnicas más complejas de programación para la elaboración de dicho trabajo final. El tema del trabajo final podrá ser alineado con los temas de investigación del alumno de posgrado, por ejemplo, vinculado a su tesis. Los asistentes también serán evaluados en su desarrollo durante los aspectos prácticos de las clases.

La evaluación del estudiante será de la siguiente manera:
La calificación final será un promedio de las notas obtenidas en la evaluación integradora y el trabajo final. Las notas de la evaluación integradora y el trabajo final deberían ser iguales o mayores a 6 (seis) para aprobar.

Modalidad de examen para estudiantes libres

- Aprobar una evaluación práctica en la semana seis (6) que abarca todo el material del curso.
- Aprobar una evaluación oral en la semana seis (6) para determinar el nivel de comprensión del material del curso.
  - La asignatura se considerará aprobada cuando se aprueben las dos evaluaciones con una nota igual o superior a 6 (seis).
- Presentar un proyecto equivalente al trabajo final de la materia.

Las notas obtenidas en cada uno de los tres puntos detallados anteriormente deberá ser igual o superior a 6 (seis). La nota final será un promedio de las tres calificaciones obtenidas.

11. Ingrese toda otra información que considere pertinente, incluidos requisitos específicos si corresponde.

1. Para que el curso sea efectivo, el profesor requerirá el uso de la sala de informática seis horas por semana, divididas en dos clases de tres horas.
2. Para enseñar el curso de manera efectiva, el profesor requerirá el uso de un proyector digital, y que Python3 y la suite Anaconda se instalen en cada una de las máquinas en la sala de computación.